2013年09月号 总第1期

51CTO.com
技术博客 Blog

IT人专业博客
领先的IT技术博客

本期热点：

- 从30岁ITer的应聘经历谈职业发展
- 中国软件开发工程师的“痛点”
- Android、iOS录音时音量大小计算

博客月刊

blog.51cto.com

目 录

从 30 岁 ITer 的应聘经历谈职业发展

作者：yaocoder 来自：http://yaocoder.blog.51cto.com/2668309/1259186

导读：ITer 的 30 问题始终是个争论，试问 30 岁低水平的 Coder 靠什么与年轻同水平的 Coder 去竞争？IT 界从来都不缺低端从业者，但是对中高级却有着强劲的需求…

自打 12 年中期第二次离职后，到这家公司已经一年有余，做的工作和所处的团队更类似于创业模式，虽然激情也多有风险。在一个多月前，团队和产品的风险已经远远高于了激情，所以在一年合同即将到期时，我更想再寻找一个更好的机会，开始自己已经处于“30 风险期”的职业生涯。

IT 人的 30 问题，始终是个争论，我一贯是鄙夷持“程序员到了 30 就无法再干下去”的论调。但是我想这句话是业界赋予了其中程序员这个关键词错误的概念，程序员这个词被完全的低端化了，试问 30 岁的低水平的 Coder 你靠什么与年轻水平的和你一样的 Coder 去竞争，你不想拿偏低的工资，由于生活问题也不能去全身心的去投入工作。IT 界包括各个行业都不缺低端从业者，但是对中高级却有着强劲的需求。

话说到自己，虽然我不承认 30 岁问题，但是现实中呢？（针对我所在的城市——济南和自我的要求来讲）从各种招聘渠道收集到的信息可以看出低端岗位居多，即使是有很多中高端岗位但是薪资却不能让自己满足。于是可选择的机会越来越少，不得不说 30 带给我更多的是生活压力问题，自我价值匹配问题，职业持续性问题，30 也给我带来了不少的压力和烦恼。

于是在找工作时选了为数不多的几个岗位，一块和大家分享下。

先看一个互联网项目研发经理岗位

第一面是和公司一个资深产品经理。我们年龄相仿，在互联网与移动互联网以及产品运营及团队管理方面进行了近两个小时的交谈，很愉快，有种志趣相投的感觉。得益于他的推荐很容易进入了下一轮人力总监的面试，和人力总监的面试就显得程序化了很多，大多数问题都是关于各种团队管理和人际处理问题，由于我管理较大团队的时间和经验有限，更多的是对以前遇到问题的总结剖析以及我的应对方法，交谈过后彼此感觉也不错，于是等着和出差还未归来的老总再一面。

再看一个创业公司高级软件工程师的岗位

这家企业是我一直关注的，因为我觉得在济南少有这样走在技术前沿的公司，有关于大数据，高性能，分布式…也是我一直渴望从事的。在去公司面试前我已经从公司网站及 linkedin 上对公司以及创始人进行了很多了解：硅谷风投、创始人和合伙人有着海外名校名企背景。很崇拜这样的公司和管理层。果然从面试就和我曾经在这个城市经历过的都不一样。

第一面，公司合伙人——王总，很活跃，很健谈，也很亲近。估计是因为在国外太久的缘故，话语中总是中英文混杂，他大量考察了我的技术，主要问的是我的海量用户承载系统的架构，技术实现。他一定是个过来人，问得非常细致，而且也有深入理解。限于

语言上的局限性，我们中间大量通过白板进行交流。交流完曾经做过的项目，他很客气的说按规矩是要再考查一下我的基础功底，于是又出了一个语言题，设计模式题，系统题，我都顺利写出。但是最后一道算法题我没答出，我承认是我的弱项，以后会强化。总之，第一面算是有险无惊的通过了。

第二面，是一个团队负责人，本来这一面的主要是来考察我的 java 水平的，(这个高级工程师的岗位主要要求有 c++经验 ,其次还要求有 java 或 python 经验) 由于我的 java 和 python 只是会使用的水平，也就没多聊这些。倒是好好聊了聊对产品设计的认识。第二面也就这样通过了，王总对我说因为 CEO-李总在美国，会对我再进行电话面试。

第三面，这是我第一次跨国的电话面试，我开始以为经历以上两轮技术面试后作为 CEO 不会再与我谈技术了，但是在长达 1 个小时的电话面试中还是主要以技术考察为主，软件设计理念和团队管理也多有涉及。结束完电面李总很客气很有礼貌的说谢谢并让我等人力通知。

再看一个偏向于外包行业的研发经理岗位

晚上下班后去的，刚到这个公司我对其印象就不好，接待人员冷漠，先让我填了一个表格，各种经历还有亲属都得写上，最后还有句承诺是真实的。接着说他们老总有事，让我稍等。等了约莫一刻钟才去老总办公室面试。典型的传统国企老总模样 ,冷淡的招呼我坐下，就开始对我的简历指指点点，说为什么跳槽这么多 (我七年，3 年第一家，3 年第二家，这个一年 ,我觉得貌似还行),然后又问我每次的离职证明有吗，我晕。接着才转向技术话题，发觉根本聊不到一块，您说您一个不懂技术的老总和我聊技术合适吗？说起管理 ,我更来气了 ,我说我的敏捷开发管理经验，他好像并不在意，中间在我说话时他吃吃药，还接电话。面试完后我们估计都对对方不满意，就算了。

这就是我的三次应聘经历，其中前两个最后已经到了谈 offer 的地步，但是由于现在公司的产品出现了转机也获得到了公司高层的一定认可，我所负责的团队也获得了相应的重视，所以并没有离开。但这三次经历确实让我对作为一个 IT 从业人员的职业发展有了不同的认识。

第一个岗位更在乎的是一个人对行业对技术的前瞻性，互联网行业是瞬息万变的，你要做好一款互联网产品除了有技术还得准确把握好行业方向，另外作为一个研发经理而言，管理应该和技术是占同样的比重的，就像他们人力总监之言，我们招的是将才，更在乎带动团队和管理团队的能力，单枪匹马走天下的人纵使再强也不合适。

第二个岗位对技术水准要求很高，但是对团队协作也颇为重视 ,采用敏捷作为团队协作方式 ,你懂的。如果想获得这种岗位你做一个仅仅只能做好本职工作的人是不行的 ,而是需要一个热爱技术 ,持续学习，主动性强的 Programmer。

第三个岗位侧重于你在一个行业的积淀，只有这样你才可以把握准用户的业务需求，和你的客户无缝的交谈。

但是，既然我们从事的是 IT 技术工作，有一点是毋庸置疑并且一定要做好的，就是做好本职工作，多学习，真正让自己的技术水准到一定高度，虽然上面有些企业并没着重考察我的技术，但是我相信我曾经的工作，或者我的博客或者我看过的书让他们跳过了这一步骤。另外，如果这次真的离职了我想我一定会选择以上第二家公司，我想那是最适合每一个热爱技术的 Programmer 的地方。

中国软件开发工程师的“痛点”

作者：李云 来自：http://yunli.blog.51cto.com/831344/1254944

作者简介：

李云，C/C++语言专家

全球著名通讯公司系统架构师，C/C++语言专家，精通嵌入式操作系统和嵌入式软件开发工作。对于面向对象开发和运行 UML 从事软件开发具有丰富的经验。擅长解决复杂的系统级软件或与硬件相关的疑难问题。

在近期的一次会议上，有高层谈到之前在中国觉得自己做得很牛，但与美国同行接触后却发现与人家存在很大的差距，这一点我在外企工作时也有过同样的体会。真正与外国同行接触后才会知道什么是差距，在这篇文章中我从软件开发工程师的角度以“痛点”的形式来谈一谈我所认为的差距。

技能之痛

相当数量的软件开发工程师（后面简称为工程师）认为除了与编码相关的内容外，其他技能都不重要。在这种意识的引导下，很容易出现的一个普遍现象是技术能力不错，但开发能力却不行。这种现象的另一种表现是：单干可以，合作不行。

技术能力是指个体对某些技术知识掌握的深度和广度，而开发能力除了包含技术能力外，还涵盖个体在项目运作过程中所需掌握的其他能力。

高效的团队一定离不开通过知识管理将个体所掌握的知识通过分享而沉淀下来。分享途径无外乎通过一定形式的文字和（或）图，这就要求工程师掌握使用象 WORD、POWERPOINT、EXCEL、VISIO（和 UML）这类工具的基本能力，并具备良好的写作与表达能力。表面看来这种能力与编码能力无关，因而也得不到工程师的普遍重视，也因此成了一个痛点。其实，写作与表达能力与编程水平息息相关，因为它们都在考验我们的逻辑思维和概念能力。忽视掌握必要工具软件的工程师难道以为编程语言是知识分享的万能工具？

个体具备良好的沟通能力是项目顺利运作的基石。不良沟通表现为：工程师在团队合作中更多采用被动询问而非主动汇报、不会辩论、对于他人指出的错误表现得“自尊”和狡辩而非感谢或承认、对于被邀请的各类审查活动（如设计审查、代码审查、文档审查）不是积极响应而需别人催促。在团队中，如果技术管理者不能很好地引导，个体沟通能力的缺乏很容易在团队中引发“一言堂”或“无政府主义”问题，工作效率低下则是必然。

专业精神之痛

不少工程师对于自己的职业缺乏精神上的追求，工作起来不求专业，只求“代码能工作就行”。这类工程师容易将经验与资历等同，以为工作年份越长就越有经验，实则不然。工作年份越长资历是越老，但如果专业水准没有在过程中不断提高的话，所获得的经验很可能趋零。

什么是专业？专业是指我们应以业内所广泛达成的共识去从事软件开发活动。这里的“业内”并非只指“国内的”，而是指“国际的”；“专业”也并到专业做事一定离不开不断地学习，只有这样才能了解行业的动向。

软件行业虽然没有“银弹”，但仍存在不少有效改善开发质量与效率的方法。只有抱着专业做事的态度去工作，我们才有可能去实践这些方法，并在实践过程中思考这些方法的内涵与不足，进而为自己的工作量体裁衣。千万不要认为“反正业内没有银弹，我要去学那么多方法干什么？”

强调专业做事的根本目的，是使我们的做事方法更科学。与我所了解的美国、俄罗斯这些国家的工程师相比，我国工程师的专业化还有很长的路要走。

速度之痛

除非你完全认可中国近些年以 GDP 为导向的经济发展策略，否则很可能得反思一下软件行业所鼓吹的“唯快不破”策略，尤其是互联网领域。
在商业环境中，“快”能获得很多竞争优势，这毋庸置疑。工程师的价值虽得（最终）体现在商业产品上，但千万不要忘记了我们始终是一名工程师，在实现商业价值的道路上不断提高自己的专业水准无论如何都不应被忘记。工程师始终要明白，公司的发展与自身的职业发展并非完全统一。如果在公司的发展过程中我们的专业水准并没有“水涨船高”，那除了说明我们在吃老本外，还表明我们很可能是在“拖后腿”。在这种情形下，即使公司蒸蒸日上地给我们发薪水，但从个体职业发展的角度说来，公司发展其实与我们“一毛钱关系都没有”。我想不致于有人认为自己以后只会在这一家公司干吧！如果真是那样想，你能保公司几十年存非单指技术内容（比如，编程语言、算法等），还包含软件项目运作中的其他各个方面（比如，开发方法、建模工具、流程、质量保证手段等）。要做在？届时万一得无奈地离开公司，单薄的专业水准又如何在人才市场与他人竞争？
对“唯快不破”的误解所带来的不良后果是，有些工程师为了快速实现软件功能而忽略了专业精神。他们一味地为了速度而筑下高额的“技术债”，甚至在“速度”的幌子下过得心安理得。

如果将“唯快不破”改为“唯效率与质量不破”或许更不容易形成误解。一说到“快”，给人的感觉往往是投入更多的时间就能达成目的，容易让人忽视做事的方法与效率。与之不同的是，强调效率需要我们考量投入时间的产出比，且暗示做事的方法只有对路才能获得效率；强调质量则提醒我们尽量别做“豆腐渣”之事，而这隐含的内容是我们必须专业做事，即使欠下了“技术债”它也时刻提醒着我们那是一定要还的。

软件行业的长期被动加班成为了速度之痛的一个缩影，它让不少工程师过着有工作没生活的日子。软件行业要避免偶尔、短期的加班是不可能的，但长期的被动加班绝对是个问题。不重视效率与质量的“勤劳”除了是在浪费外，更是一种透支将来的短视行为。

视野之痛

视野之痛体现在工程师在从事技术工作时，忽视了解国外的发展状况。他们因为不知道同质开源项目的存在而走上“重新发明轮子”的道路，甚至发明出“三角形的轮子”；也因为对英文资料缺乏阅读的耐心而不去了解相关国际标准、订阅开源项目的 MAILING LIST 和专业网站的 NEWSLETTERS 等。

狭窄的视野很容易让人自满，以为软件开发就是那么简单，最后导致成长慢、意识与技能“不入流”。

以我的经验来看，工程师如果不能很好地阅读英文资料则要达到高技术水平实在很难，视野狭窄也成必然。另外，编程活动中的命名环节其实对我们的英语水准提出了一定的要求，不然很容易动名词不分而写出只有自己容易读懂的程序，或常出现命名时找不到合适的单词去精确表达程序意图。

持续发展之痛

以上各痛点的最终结果又给我们带来了持续发展之痛。其表现为：少有人会在项目中通过文档提升开发效率；鲜有人会持续改善软件的设计质量；大部分人只关注短期完成工作，而忽视短期行为所带来的高额隐性成本。

持续发展之痛使得工程师很难轻装上阵，工作精力过多花费在重复、低级的琐事上，而非用于学习和思考。最终结果是将工作变成了“青春饭”，辛苦但却看不到美好的未来。

所有痛点可以归结为意识的陈旧，或虽有意识却无力于将其转变为能力！（注：意识是一种行为，而非能力）

当然，这些“痛”与我国的社会大环境有着紧密的联系。但无论是怎样的环境，总有人做得出色，或许他们身上有我们所没有的内容。是什么？只有自己去想、去悟，成长之痛！ 即使大环境好了、大家都很“专业”，职场的“金字塔”总是摆在那的。谁能向上走？走多远？全靠个人，没有SHORTCUT！只不过每个人都平等地拥有向上走的机会与权力！

博主相关文章推荐：

1. 通讯与互联网行业软件项目运作有哪些不同？

   http://yunli.blog.51cto.com/831344/1216369

2. 【Chrome】RSA 算法在扩展程序中的运用

   http://yunli.blog.51cto.com/831344/1211260

3. 软件质量管理之困境与对策思考

   http://yunli.blog.51cto.com/831344/1060117

Android、iOS 录音时音量大小计算

作者：何金来 来自：http://ikinglai.blog.51cto.com/6220785/1256781

经常有人问我如何计算录音时音量大小。iOS 平台是有 API 可以直接调用的，但是 Android 平台上没有比较好的办法，因此我们就不得不自己计算了。

之所以有计算音量这个需求，是因为很多应用希望根据音量的大小实现一些动画效果。因此，从这个需求出发，只要能根据说话时声音的大小，获取到的音量值有变化即可，而不必过分纠缠于到底范围多少才是准确的。因为计算的方法有很多种，不同的方法计算出来的值肯定是不同的，但是只要能反映出大小变化，我们的目的就达到了。

下面我以 Android 录音为例，介绍一下其中的一种计算方法。大家可以根据自己的需要重新计算，或者是对我这个计算出来的值做一些数学变换，从而满足自己的需要。

需要提前说明的是，网上也有类似的一些计算方法，但是千万不能照搬过来，因为这个和录音的编码和录音数据的类型是有关系的。

录音的编码主要有两种：8 位 pcm 和 16 位 pcm。8 位 pcm 用一个字节表示语音的一个点，16 位 pcm 用两个字节，也就是一个 short 来表示语音的一个点。需要特别注意的是，如果你用的 16 位 pcm 编码，而取录音数据用的是 byte 的话，需要自己将两个 bye 转换成一个 short。将两个 byte 转换成一个 short，有小端和大端两种，一般默认情况都是小端，但是有的开源库，比如 lamemp3 需要的就是大端，这个要根据不同的情况进行不同的处理。

下面以 Android 为例，介绍一下用平均值计算音量的方法。

```
private double calculateVolume(short[] buffer){
        double sumVolume = 0.0;
        double avgVolume = 0.0;
        double volume = 0.0;
        for(short b : buffer){
                sumVolume += Math.abs(b);
        }
        avgVolume = sumVolume / buffer.length;
        volume = Math.log10(1 + avgVolume) * 10;
        return volume;
```

这个方法传递的是 short 类型的数据，所以录音的编码肯定是 16 位 pcm，这样可以直接计算而不需要转换了。相信大家都听过声波这个东西，大家用音频编辑软件 Adobe audition 打开一段声音：

从这里我们可以看到，声音是高低起伏变化的，有波峰波谷，说白了就是有正有负。因此在计算的时候，我们需要先求绝对值，要不然就上下抵消。求完绝对值然后进行累加，再除以整个数据的长度，就得到了这段语音数据的平均值了。

但是这样直接计算出来的结果比较大，不利于我们使用，因此对它进行了取对数再乘以 10：

```
volume = Math.log10(1 + avgVolume) * 10;
```

这些可以根据自己的需要进行运算，我这边只是一个简单的例子。

还有一个特别需要注意的问题是：如果你录音的编码是 16 为 pcm，而录音数据数据是 byte，需要将两个 byte 转为一个 short 进行处理，建议用小端的方式。

```
private doublecalculateVolume(byte[] buffer){
        double sumVolume = 0.0;
        double avgVolume = 0.0;
        double volume = 0.0;
        for(int i = 0; i < buffer.length; i+=2){
                int v1 = buffer[i] & 0xFF;
                int v2 = buffer[i + 1] & 0xFF;
                int temp = v1 + (v2 << 8);// 小端
                if (temp >= 0x8000) {
                        temp = 0xffff - temp;
                }
        sumVolume += Math.abs(temp);
        }
        avgVolume = sumVolume / buffer.length / 2;
        volume = Math.log10(1 + avgVolume) * 10;
        return volume;
}
```

关于用小端的方式，将两个 byte 转为一个 short 的那些移位运算，应该不难，这里不做过多的解释了。

以上就是用平均值的方法计算音量大小的方法，但是在 iOS 中，你自己不需要计算了，直接调用系统的 api 即可。

iOS 获取录音是系统音量的步骤：

1、在开始录音之前，要设置获取系统音量的属性

```
// 设置获取音量大小的属性
  UInt32 enabledLevelMeter = true;
  AudioQueueSetProperty(_audioQueue,
kAudioQueueProperty_EnableLevelMetering,
&enabledLevelMeter,sizeof(UInt32));
```

2、调用系统 api 获取音量，此方法可以在 缓冲器的回调函数中调用，也可以自己用一个定时器来调用

```
AudioQueueLevelMeterState levelMeter;
   UInt32 levelMeterSize = sizeof(AudioQueueLevelMeterState);
  AudioQueueGetProperty(_audioQueue,
kAudioQueueProperty_CurrentLevelMeterDB, &levelMeter, &levelMeterSize);
    double volume = levelMeter.mAveragePower
```

当然，你如果不想用系统的音量值，也可以像上面那样自己计算，这取决于你的需要。

相关技术文章推荐：

1. Android 应用程序资源的编译和打包过程分析

   http://shyluo.blog.51cto.com/5725845/1229261

2. Dalvik 虚拟机进程和线程的创建过程分析

   http://shyluo.blog.51cto.com/5725845/1229256

3. Android 应用程序资源的查找过程分析

   http://shyluo.blog.51cto.com/5725845/1229259

DAS,NAS,SAN 在数据库存储上的应用

作者：张骞 来自：http://qianzhang.blog.51cto.com/317608/1254617

硬盘接口类型

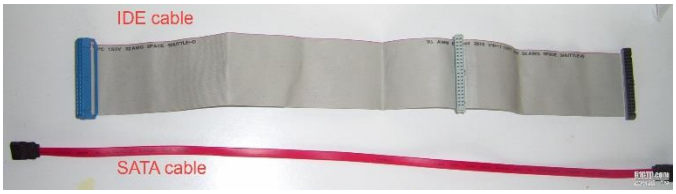
并行接口还是串行接口

并行接口，指的是并行传输的接口，比如有 0~9 十个数字，用 10 条传输线，那么每根线只需要传输一位数字，即可完成。

从理论上看，并行传输效率很高，但是由于线路上的物理原因，它的传输频率不能太高，所以实际的传输速度并不和并行度成正比，甚至可能更差。

串行接口，指的是串行传输的接口，同样是 0~9 十个数字，用 1 条传输线，那么需要传输 10 次，才可以完成。从理论上看，串行传输效率不高，但是由于它的数据准确性，高频率的支持，使得传输速度可以很高。

并行连接线和串行连接线(IDE/SATA)



目前，计算机的外部接口大多被串行接口取代，比如：USB，1394，COM 等等，都是串行接口。而硬盘的外部接口也已经被串行接口(SATA/SAS)占领。

硬盘接口类型

按硬盘接口协议/规范可以分为 2 类：ATA 和 SCSI。使用了某种接口的硬盘就称为 XXX 硬盘。

ATA 接口协议

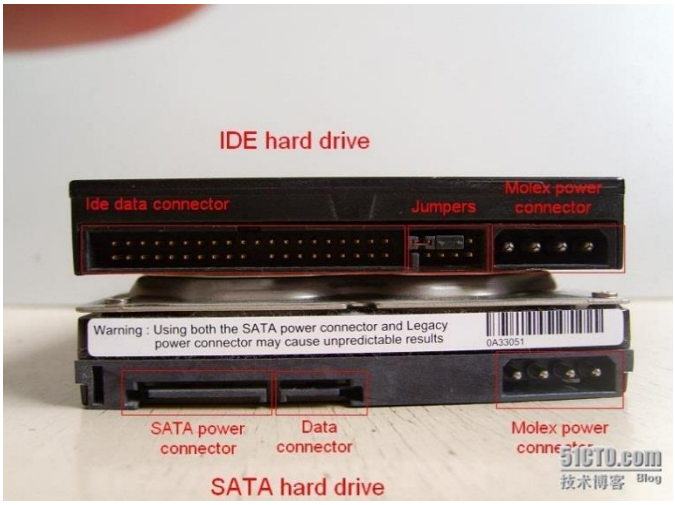
IDE 接口

IDE 接口也称为 PATA(Parallel ATA)接口，也就是并行 ATA 接口。以前的 PC 机大多用的这种接口的硬盘。

SATA 接口

SATA(Serial ATA)接口，串行 ATA 接口，这类硬盘，转速通常不太高，容量大，目前 PC 机或者 IOPS 要求不是太高的存储多使用这种接口的硬盘。

IDE 和 SATA 硬盘接口图示



SCSI 接口协议

SCSI 接口

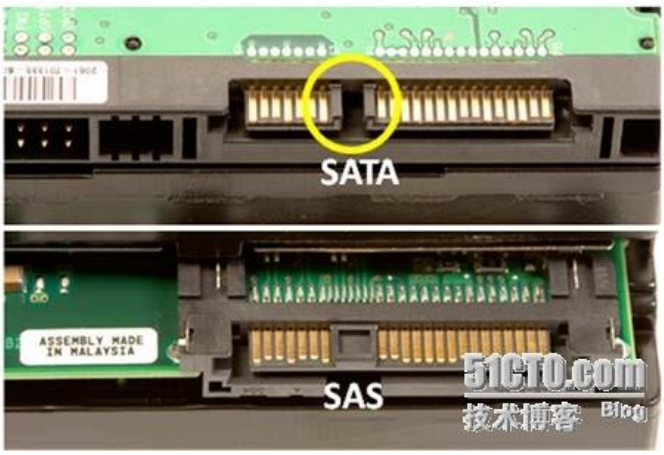
通常所说的 SCSI，是一种并行接口，早期的计算机外设

(打印机，扫描仪等等)，也大多使用这种接口。现在使用这种接口的硬盘已经很少。

SAS 接口

SAS(Serial SCSI)接口 ,串行 SCSI 接口 ,这类硬盘，转速高，IOPS 高，适用于 OLTP 系统的存储。

另外，SAS 的接口技术已经可以兼容 SATA，也就是说：如果主板上有个 SAS 接口，是可以接 SATA 硬盘的，但是反之不行。从图片上看，SAS 和 SATA 接口有点相似。



存储方案

所谓的存储方案，就是用单独的软硬件将磁盘/磁盘组管理起来，供主机使用。

目前的外挂存储解决方案主要分为三种：

(1) 直连式存储（DAS：Direct Attached Storage)

(2) 网络存储设备（NAS :Network Attached Storage)

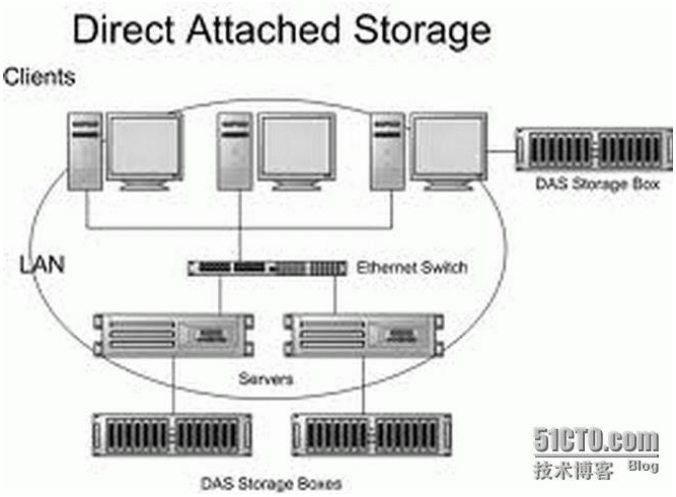
(3) 存储网络（SAN：Storage Area Network)

存储方案内部使用的硬盘，多为 SATA/SAS，经过串联/RAID 之后，对主机提供访问接口。

DAS

直接连接存储（DAS:Direct Attached Storage)，是指将存储设备通过 SCSI 接口或 FC 接口直接连接到一台计算机上。DAS 不算是网络存储，因为只有它所挂载的主机才可访问它。

也就是说，服务器发生故障时，连接在服务器上的 DAS



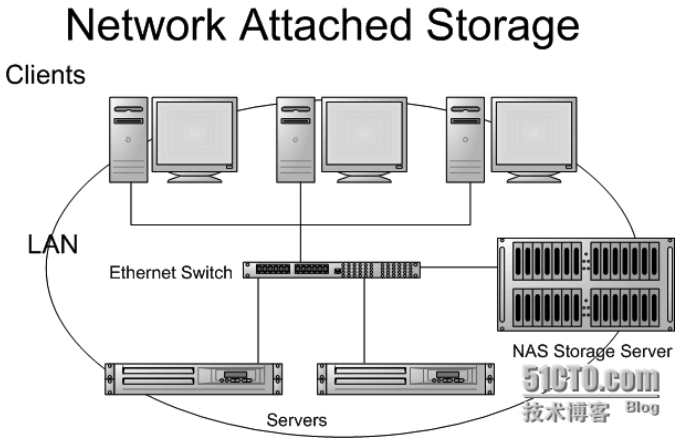
存储设备中的数据暂时不能被存取。

DAS 设备示例：

NAS

网络连接存储（NAS:Network Attached Storage)，是指将存储设备通过标准的网络拓扑结构（例如以太网），连接到一群计算机上。NAS 有文件系统和 IP 地址，可以类似的理解为网上邻居的共享磁盘。

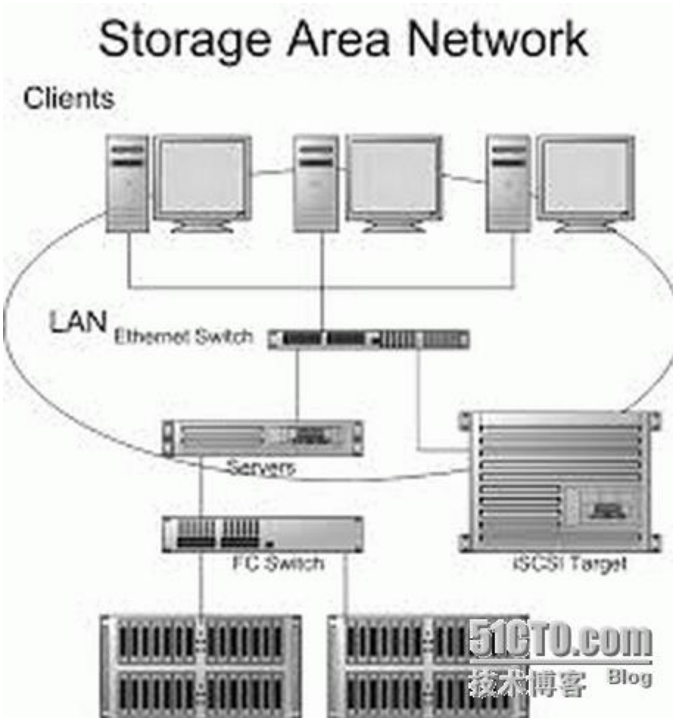
NAS 设备示例：



SAN

存储区域网络(SAN：Storage Area Network)，目前的 SAN 存储有 2 种：一是基于光纤通道的 FC SAN；二是基于以太网的 IP SCAN(也就常说的 iSCSI)。

FC SAN 通过光纤交换机连接到主机(HBA 卡)，也就是说可以连接到光纤交换机的主机都可以访问这个存储；iSCSI 作为共享于以太网络上的存储则更类似于 NAS。



FC SAN 设备示例：

iSCSI 设备示例：

在数据库存储上的应用

三种存储方案的比较，如图：

DAS 、NAS 和 SAN 比较

	DAS	NAS	SAN
安装	简单	十分简单，以分钟计算	比较麻烦，以天计算
连接介质	SCSI 或光纤	网络线缆	SCSI 或光纤
传输协议	SCSI	TCP/IP	FCP （光纤通道协议）
数据传输类型	SCSI 数据块	文件级	SCSI 数据块
可扩展性	差	好	好
数据集中化	否	是	是
数据传输速率	较高	较低	高
覆盖地理范围	较小	广	较广
集群应用支持	较好	差	好
数据库支持	好	差	好
多平台支持	不支持	支持	通过软件可实现磁盘共享
维护成本	高	底	较高
价格	低	高	较高
兼容性	较好	标准协议、高	标准不统一、差

由于NAS自身是以文件为传输单元，而数据库是以数据块为传输单元，在大型数据库上，如果想达到较高的性能，不建议使用NAS产品。

DAS

可作为本机的外挂硬盘，不过现在单块磁盘的空间已经很大，如果几个 T 的空间，直接在主机里插硬盘就

可以实现了，不需要外挂。

NAS

由于它的文件系统特性，加上以太网网线传输，更像是我的电脑-网上邻居-共享磁盘 ,访问方式也是类似：\\NAS01\BACKUP\database_name.bak。更多的是作为文件共享、备份、归档所用，比如数据库的历史备份/异地备份文件。

SAN

FC SAN 使用光纤传输，是一个高速的共享存储，数据库的任何东西都可以放在上面，还有就是在做集群时(failover clustering) 作为仲裁盘；

iSCSI 的传输速率要低于 FC SAN，目前在我们的环境中还没有直接使用 iSCSI 做数据库存储，通常是用在类似 NAS 的地方。

相关文章推荐：

1. RAID 在数据库存储上的应用
   http://qianzhang.blog.51cto.com/317608/1251260
2. 磁盘读写与数据库的关系
   http://qianzhang.blog.51cto.com/317608/1249534

Office 程序出错的几种原因与解决方法

作者：吕卓贤 来自：http://nearlv.blog.51cto.com/2432295/1252414

Office 程序出错的几种原因与解决方法

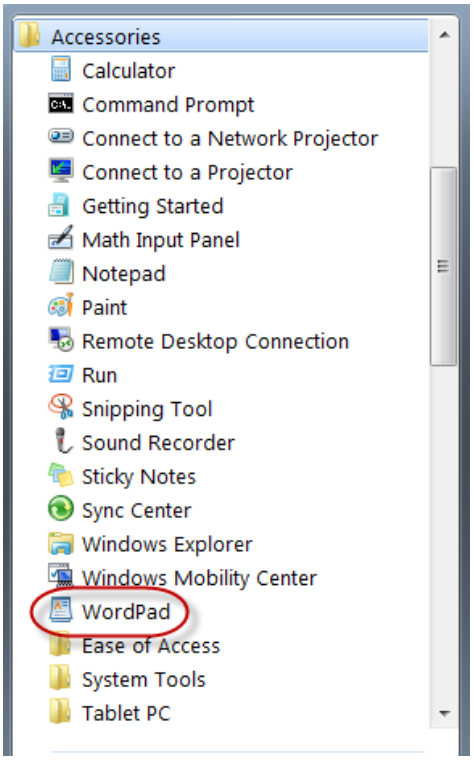
以下的几种情况都是最近发生在公司电脑的问题，困扰着很多的同事，之前一直都是直接重新安装 Office 程序，虽然能够解决问题，但是费时又费力，特在这里分享一下具体的情况与解决方法

问题一：文档提示损坏，转换出错

分析：这类出错文档一般都是发生格式为“.rtf” or “doc”之类的 WORD 文档，Excel 程序也会出现类似的问题，我们知道，如果使用 Office 2007 版以下的版本去打开“.docx”,”.xlsx”这类文档，程序需要安装一个转换工具，而这种错误就是发生在这个转换的过程无法进行下去，所以导致文档出错。

解决方法：使用系统的 WordPad（它是微软系统的一个文本编辑器，大家有兴趣可以搜索一下相关资料）去修复出现此问题的文档，操作方法如下：

1）、从开始菜单-所有程序-附件下找到 WordPad，如下：



2）、通过 WordPad 文件菜单下选择“打开”，在出现的窗口当中选择出现问题的文档，重新保存一份后缀为”.docx”的文档。

要点：这就有点像有时我们不知道一个文件应该用什么程序去打开它，比如 Flash 文件，我们一般会想到用 Flash Player 去打开，但其实也是可以用 IE 浏览器去打开它，换位思考；

问题二、宏的安全设置问题

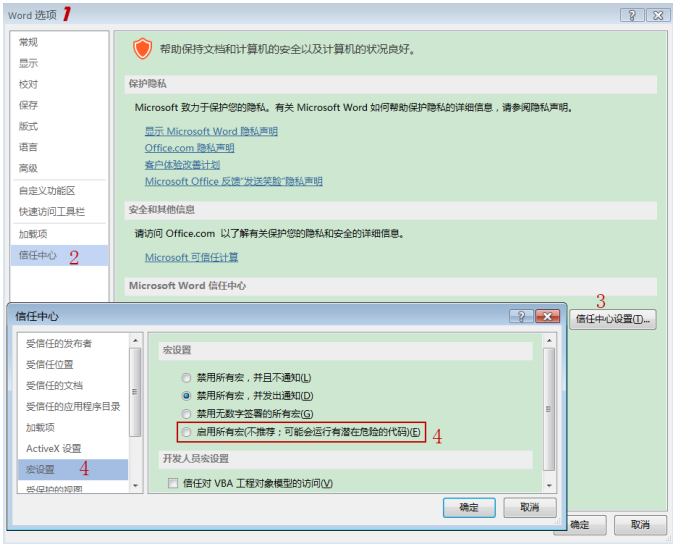
在 Word 和 Excel 程序环境下出现这种问题的概率比较多，但它也会影响其它的 Office 程序.

解决方法：修改宏的安全设置

Office 程序默认是禁用宏的运行的，比如我们在工作当中，Excel 使用宏的情况是非常多的，宏能给我们带来很大的帮忙，特别是涉及到一些数据库链接的情况，方便的同时也存在身一定的安全隐患，下面我们一起看如何取消它

1）、启动 Word / Excel 程序；

2）、依次打开文件—选项—信任中心-信任中心设置-宏设置，如下：



3） 将“启用所有宏（不推荐；可能会运行有潜在危险的代码）”前面的选项打上勾，然后确定所有的设置，再去打开有问题的文档；

补充：通过系统的添加安装程序找到系统更新，留意一下最新更新了什么补丁，如果有，则进行删除并测试程序是否还会出错。

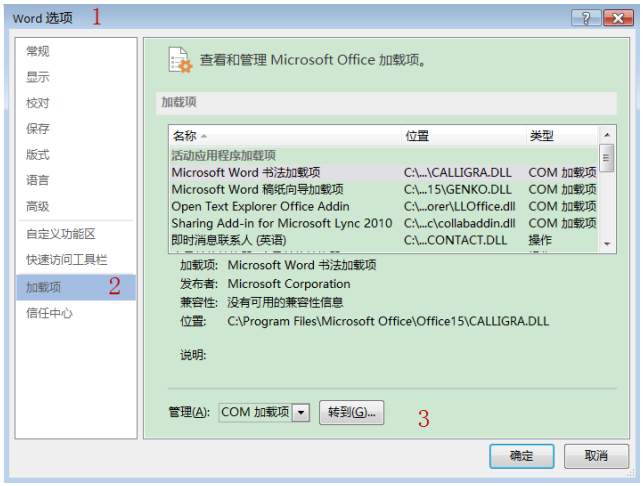
问题三：蓝牙 Add-in 的问题

启用蓝牙的这个加载项有可能会出现 Office 程序出现，这种情况大概发生在使用了系统更新后而出现的问题

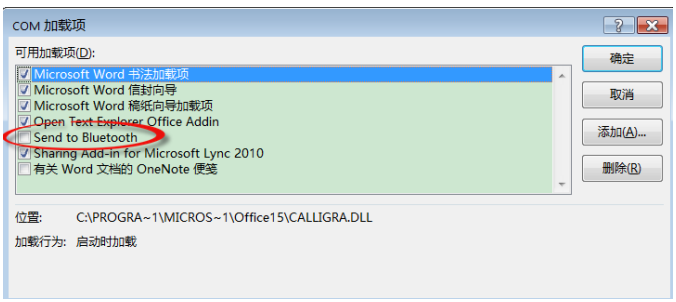
解决方法：

1）启动 Word / Excel 程序

2）选择文件选项卡—选项---加载项---管理：Add-ins，单击“转到”，如下：



3）出现如下对话框，将“Send to Bluetooth”前面的选项勾去掉，然后点击确定



4）如果程序还是继续出错，你可能还要按照这个方法去设置一下 Excel, PowerPoint 等。

解决方法二：启用程序的安全模式

程序的安全模式下是不加载任何的加载项，如果安全模式能够正常运行，那基本上可以判定是加载项的问题，这将缩短您解决问题的范围。

相关文章推荐：

1. Vlookup 跨表查询实战演示
   http://qianzhang.blog.51cto.com/317608/1251260
3. 磁盘读写与数据库的关系
   http://qianzhang.blog.51cto.com/317608/1249534

通过 Lua 解释器来扩展丰富 nginx 功能，实现复杂业务的处理

作者：芮峰云 来自：http://rfyiamcool.blog.51cto.com/1030776/1248825

Nginx 很强

Lua 很强

所以，他俩和一起也一定很强。。。（多么霸道的逻辑呀~）

Lua_Nginx_Module 可以一步步的安装，也可以直接用淘宝的 OpenResty

Centos 和 Debian 的安装就简单了。。

这里说下 Freebsd 的安装：

```
fetch http://www.lua.org/ftp/lua-5.1.4.tar.gz
tar zxvf lua-5.1.4.tar.gz
cd lua-5.1.4
make freebsd
make install
cd ..
fetch https://github.com/chaoslawful/lua-nginx-module/zipball/v0.1.6rc2
fetch https://github.com/simpl/ngx_devel_kit/zipball/v0.2.17rc2
tar zxvf v0.1.6rc2
mv chaoslawful-lua-nginx-module-ccaf132 lua_nginx_module
tar zxvf v0.2.17rc2
mv simpl-ngx_devel_kit-bc97eea ngx_devel_kit
tar zxvf pcre-8.12.tar.gz
tar zxvf nginx-1.0.3.tar.gz
cd nginx-1.0.3
./configure --prefix=/data/soft/nginx --with-pcre=../pcre-8.12 --add-module=../ngx_devel_kit -
-add-module=../lua_nginx_module
make && make install
```

Ok 后，说下 Lua 这个强大的玩意...

简单的输出一句话...

Ngx.Say 是打印的打印输出的意思...

```
location /echo {
default_type text/plain;
echo hello lua;
}
location /lua {
default_type text/plain;
content_by_lua 'ngx.say("hello world")';
}
```

访问的限制...

```
location @client{
proxy_pass  http://www.ruifengyun.com;
}
location ~  /test {
default_type  text/html;
content_by_lua 'ngx.say("this is  ruifengyun.com!")';
access_by_lua '
if ngx.var.remote_addr == "10.2.20.110" then
ngx.exit(ngx.HTTP_FORBIDDEN)
end
if ngx.var.remote_addr == "10.2.20.112" then
ngx.exec("@client")
end
';
}
```

控制经过判断之后，才能访问

```
location / {
access_by_lua '
local res = ngx.location.capture("/auth")
if res.status == ngx.HTTP_OK then
return
end
if res.status == ngx.HTTP_FORBIDDEN then
ngx.exit(res.status)
end
ngx.exit(ngx.HTTP_INTERNAL_SERVER_ERROR)
';
# proxy_pass/fastcgi_pass/postgres_pass/...
}
```

rewrite 跳转

这个是先判断 check-pam 接口的 return 的内容是不是 spam，是的话，转跳到其他的页面

```
location / {
rewrite_by_lua '
local res = ngx.location.capture("/check-spam")
if res.body == "spam" then
ngx.redirect("/terms-of-use.html")
end
'; fastcgi_pass ...;
}
```

一个根据 ip 做匹配

```
location / {
content_by_lua '
            myIP = ngx.req.get_headers()["X-Real-IP"]
            if myIP == nil then
                myIP = ngx.req.get_headers()["x_forwarded_for"]
            end
            if myIP == nil then
                myIP = ngx.var.remote_addr
            end
            if myIP == "" then
                ngx.exec("@client")
            else
                ngx.exec("@client_test")
            end
        ';
}
```

redirect 的使用

```
return ngx.redirect("/foo")
return ngx.redirect("http://localhost:1984/foo", ngx.HTTP_MOVED_TEMPORARILY)
return ngx.redirect("/foo", 301)
返回302临时重定向 地址栏会显示跳转后的地址
rewrite ^ /foo? redirect;  # nginx config
return ngx.redirect('/foo');  -- lua code
```

过滤 post 过来的参数

```
location = /test {
content_by_lua '
ngx.req.read_body()
local args = ngx.req.get_post_args()
for key, val in pairs(args) do
if type(val) == "table" then
ngx.say(key, ": ", table.concat(val, ", "))
else
ngx.say(key, ": ", val)
end
end
';
}
```

一个 Lua 的例子：

```
#!/usr/bin/env lua
ngx.say('aaaaaa </br>')
local url = ngx.var.uri
ngx.say('<br>',url,'<br/>')
ngx.print('这次访问的header头是  ',ngx.req.raw_header())
ngx.print('<meta http-equiv="content-type" content="text/html;charset=utf-8">')
ngx.print('<h1> 这个是 h1 </h1>')
ngx.print('这次访问的是 get 还是 post 呀  ',ngx.req.get_Method())
local args = ngx.req.get_uri_args()
ngx.print(args)
local res = ngx.location.capture("/")
ngx.print('<br>http code <br>',res.status)
```

一个 mysql 的例子，from 春哥

```
worker_processes  2;
error_log logs/error.log warn;
events {
worker_connections 1024;
}
http {
upstream backend {
drizzle_server 127.0.0.1:3306 protocol=mysql
dbname=ngx_test user=ngx_test password=ngx_test;
drizzle_keepalive max=10 overflow=ignore mode=single;
}
server {
listen 8080;
location @cats-by-name {
set_unescape_uri $name $arg_name;
set_quote_sql_str $name;
drizzle_query 'select * from cats where name=$name';
drizzle_pass backend;
rds_json on;
}
location @cats-by-id {
set_quote_sql_str $id $arg_id;
drizzle_query 'select * from cats where id=$id';
drizzle_pass backend;
rds_json on;
}
location = /cats {
access_by_lua '
if ngx.var.arg_name then
return ngx.exec("@cats-by-name")
end
if ngx.var.arg_id then
return ngx.exec("@cats-by-id")
end
';
rds_json_ret 400 "expecting \"name\" or \"id\" query arguments";
}
}
}
```

改改密码就能用啦~

获取 url 中的参数

```
location = /adder {
    set_by_lua $res "
            local a = tonumber(ngx.arg[1])
                local b = tonumber(ngx.arg[2])
                return a + b" $arg_a $arg_b;

        echo $res;
}
```

ngx.req.set_uri

nginx 里面的配置是：

```
location /test {
        rewrite ^/test/(.*) /$1 break;
        proxy_pass http://my_backend;
    }
```

lua 里面的配置是：

```
location /test {
      rewrite_by_lua '
          local uri = ngx.re.sub(ngx.var.uri, "^/test/(.*)", "$1", "o")
          ngx.req.set_uri(uri)
      ';
      proxy_pass http://my_backend;
  }
```

我想大家看这个对照，已经知道是啥意思了.

通过 lua 获取 nginx 的内置变量，通过这些变量做些逻辑的处理~

```
Nginx提供了很多内置的变量,如:
$arg_PARAMETER  这个变量包含在查询字符串时GET请求PARAMETER的值。
$args  这个变量等于请求行中的参数。
$binary_remote_addr  二进制码形式的客户端地址。
$body_bytes_sent  传送页面的字节数
$content_length  请求头中的Content-length字段。
$content_type  请求头中的Content-Type字段。
$cookie_COOKIE  cookie COOKIE的值。
$document_root  当前请求在root指令中指定的值。
$document_uri  与$uri相同。
$host  请求中的主机头字段，如果请求中的主机头不可用，则为服务器处理请求的服务器名称。
$is_args  如果$args设置，值为"?"，否则为""。
$limit_rate  这个变量可以限制连接速率。
$nginx_version  当前运行的nginx版本号。
$query_string  与$args相同。
$remote_addr  客户端的IP地址。
$remote_port  客户端的端口。
$remote_user  已经经过Auth Basic Module验证的用户名。
$request_filename  当前连接请求的文件路径，由root或alias指令与URI请求生成。
$request_body  这个变量（0.7.58+）包含请求的主要信息。在使用proxy_pass或fastcgi_pass指令的location中比较有意义。
$request_body_file  客户端请求主体信息的临时文件名。
$request_completion  未知。
$request_method  这个变量是客户端请求的动作，通常为GET或POST。
包括0.8.20及之前的版本中，这个变量总为main request中的动作，如果当前请求是一个子请求，并不使用这个当前请求的动作。
$request_uri  这个变量等于包含一些客户端请求参数的原始URI，它无法修改，请查看$uri更改或重写URI。
$scheme  所用的协议，比如http或者是https，比如rewrite  ^(.+)$  $scheme://example.com$1  redirect;
$server_addr  服务器地址，在完成一次系统调用后可以确定这个值，如果要绕开系统调用，则必须在listen中指定地址并且使用bind参数。
$server_name  服务器名称。
$server_port  请求到达服务器的端口号。
$server_protocol  请求使用的协议，通常是HTTP/1.0或HTTP/1.1。
$uri  请求中的当前URI(不带请求参数，参数位于$args)，可以不同于浏览器传递的$request_uri的值，它可以通过内部重定向，或者使用index指令进行修改。
另外:
HTTP_X_FORWARDED_FOR是通过代理服务器取得客户端的真实IP地址，有些用此方法读取到的仍然是代理服务器的IP。还有一点需要注意的是：如果客户端没有通过代理服务器来访问，那么用 HTTP_X_FORWARDED_FOR 取到的值将是空的。
```

函数版的访问

```
location /lua1 {
default_type 'text/plain';
content_by_lua 'ngx.say("hello, lua")';
}
# 请求另外的url
location /lua2 {
content_by_lua '
    local res = ngx.location.capture("/hello1")
    ngx.say("data: " .. res.body)
';
}
```

扫描二维码，加 51CTO 博客为朋友

关注 51CTO 博客微信，打造你的个性！

http://blog.51cto.com

编后语

《51CTO 博客月刊》为 51CTO 博客整理出品

如果您有意投稿，请联系我们
如果您有反馈意见，请告诉我们

联系方式：
Email：blog@51cto.com
QQ: 1173854158

欢迎关注我们：
@51CTO 技术博客　　@中国 IT 博客大赛